BUNDESREPUBLIK DEUTSCHLAND

Prioritätsbescheinigung über die Einreichung einer Patentanmeldung

Aktenzeichen:	198 46 676.5
Anmeldetag:	09. Oktober 1998
Anmelder/Inhaber:	Siemens Aktiengesellschaft, München/DE
Bezeichnung:	Verfahren zur Absicherung von Einsprungsadressen
IPC:	G 06 F 9/42

Die angehefteten Stücke sind eine richtige und genaue Wiedergabe der ursprünglichen Unterlagen dieser Patentanmeldung.

München, den 22. März 2001
Deutsches Patent- und Markenamt
Der Präsident
Im Auftrag

Nietiedt

A 9161
06/00
EDV-L

CERTIFIED COPY OF PRIORITY DOCUMENT

Beschreibung

Verfahren zur Absicherung von Einsprungsadressen

Auf zukünftigen Chipkarten sollen Applikationen von Drittfirmen zugelassen werden, die Funktionen des Betriebssystems aufrufen können. Dabei könnte die Gefahr bestehen, daß diese Applikationen von Drittfirmen Manipulations- oder Sabotageversuche hinsichtlich anderer Programmteile enthalten. Ein solcherart möglicher Angriff wäre, nicht einen gültigen Funktionszeiger einer Betriebssystemroutine aus der Applikation heraus zu verwenden, sondern eine andere Einsprungsadresse zu verwenden. Dadurch würde Code des Betriebssystems nicht ordnungsgemäß ausgeführt und z.B. möglicherweise Datenverluste durch fehlerhaftes Überschreiben eines Speicherbereiches provoziert werden.

Gemäß dem Stand der Technik ist eine Lösung durch den Einsprung auf vordefinierte Adressen in festen Abständen (gates) geplant. Dabei muß zunächst überprüft werden, ob eine Gateadresse im Bereich der für das Modul erlaubten Adressen liegt. Bei positivem Resultat wird das gate angesprungen und von dort weiter ein Sprung in die eigentliche Funktion ausgeführt. Dies ist nachteilig, da dabei ein berechneter Sprung ausgeführt werden muß, der zu einem Peephole in der CPU Pipeline führt. Alternativ kann der Sprung in die Funktion verzögert nach einigen vorgezogenen Programminstruktionen der Funktion stattfinden. Dies führt zu einem schwierigen Optimierungsproblem für den Compiler, falls am Anfang der Funktion eine Schleife eingeleitet wird und ständig zwischen dem Code am gate und dem ausgelagerten Code hin und her gesprungen werden muß.

Es ist daher Aufgabe der Erfindung, ein Verfahren zur Absicherung von Einsprungsadressen dieser Art zu schaffen, bei dem kein berechneter Sprung ausgeführt werden muß und damit kein Peephole in der Pipeline der CPU-Instruktionen auftritt.

Erfindungsgemäß wird diese Aufgabe dadurch gelöst, daß zulässige Einsprungsadressen direkt angesprungen werden dürfen, und durch eine Korrelation von Daten, die nicht innerhalb derselben, einzelnen Instruktion stehen können, erkennbar sind.

Dabei kann der Compiler oder Linker durch Organisation des Programmcodes sicherstellen, daß nur legale Einsprungsadressen diese Korrelation erfüllen. Beispielsweise kann die Korrelation dadurch erfolgen, daß die Speicherzelle unmittelbar vor oder nach der Einsprungsadresse die Adresse korrelierter Daten enthält.

Eine bevorzugte Möglichkeit ist dabei, daß die Speicherzelle unmittelbar vor oder nach der Einsprungsadresse einen Verweis auf den entsprechenden Eintrag in einer geschützten Liste legaler Einsprungsadressen enthält.

Es ist dabei besonders bevorzugt, beim Ausführen des Funktionsaufrufs automatisch zu überprüfen, ob die Korrelation der Daten erfüllt ist.

Besonders bevorzugt ist es, beim Ausführen des Funktionsaufrufs zusätzlich automatisch zu überprüfen, ob das korrelierte Datum im vorgesehenen, reservierten Speicherbereich liegt.

Sofern Programminstruktionen eine gewisse maximale Anzahl n von Bytes nicht überschreiten, kann eine weitere erfindungsgemäße Lösung angewendet werden, indem ein spezieller no-operation Code vorgesehen wird, der zur Vermeidung zufälliger Korrelationen dient, und vom Compiler oder Linker nachträglich eingefügt werden kann.

Dabei ist es besonders bevorzugt, daß die Korrelation zwischen Codedaten erfolgt, die mindestens n Byte voneinander entfernt sind.

Außerdem kann die Einsprungsadresse erfindungsgemäß durch das Einfügen einer speziellen Bytesequenz abgesichert werden, die nicht innerhalb des regulären Codes auftreten kann, beispielsweise durch einen speziellen no-operation Code.

Erfindungsgemäß kann also das Peephole in der Pipeline durch einen direkten Ansprung an die Adresse des Funktionspointers vermieden werden. Allerdings muß dann dafür gesorgt werden, daß legale Ansprungsadressen durch nichtlokale Codekorrelationen ausgezeichnet sind. Der Compiler oder Linker muß durch Organisation des Programmcodes sicherstellen, daß nur legale Funktionsansprungadressen diese Korrelation erfüllen. "Nichtlokale Korrelation" bedeutet hierbei eine Korrelation von Daten, die nicht innerhalb der selben, einzelnen Instruktion stehen können.

Folgende bevorzugte Ausführungsformen der Erfindung sind also beispielsweise möglich:

1. Korrelation mit Daten in dafür reservierten Speicherbereichen: Eine einfache Implementation könnte z.B. in der Speicherzelle unmittelbar vor der Einsprungsadresse die Adresse eines korrelierten Datums beinhalten, das z.B. wieder der legalen Einsprungsadresse der Funktion entspricht. Beim Ausführen des Funktionsaufrufs kann automatisch überprüft werden, ob diese Korrelation erfüllt ist und/oder ob das korrelierte Datum im vorgesehenen, reservierten Speicherbereich liegt. Der Mechanismus ist auf den ersten Blick sehr ähnlich zum bisherigen Gatemechanismus, hat aber den Vorteil, daß kein berechneter Sprung auftritt und die Instruktionen der Funktion unmittelbar in den Prefetcher der Pipeline geholt werden können. Ein Peephole tritt nur im Fehlerfall eines illegalen Einsprungs auf.

2. Korrelation mit Programmdaten in nicht reservierten Speicherbereichen. Diese Lösung setzt voraus, daß die Programminstruktionen eine gewisse maximale Anzahl n von Bytes nicht

überschreiten. Längere Datenbereiche im Codesegment müssen dann ausgeschlossen werden. Außerdem setzt diese Methode einen speziellen no-operation Code (SNOP) voraus, der niemals im regulären Code eingesetzt wird und nur zum Vermeiden von zufälligen Korrelationen vom Compiler/Linker nachträglich eingefügt wird. Zwei unterschiedliche Typen von Lösungen lassen sich hier noch unterscheiden:

a) Die Korrelation erfolgt zwischen Codedaten, die mindestens n Bytes voneinander entfernt sind. Der Compiler bzw. Linker muß dabei sicherstellen, daß etwaige zufällige Korrelationen im Code durch Einführen von SNOP-Zwischencodes vermieden werden.

Eine mögliche Implementation sieht folgendermaßen aus: Unmittelbar vor der Einsprungsadresse der Funktion steht ein Wert, der eine Funktion der darauf folgenden n+m ($m \geq 0$, ansonsten beliebig) Bytes ist. Sollte diese Korrelation irgendwo im Code zufällig erfüllt sein, so muß der Compiler bzw. Linker diese zufällige Korrelation aufheben: Da in der Folge der n+m Bytes nach Voraussetzung mindestens eine echte Instruktion endet, kann nach deren Ende eine Reihe von SNOP-Instruktionen eingefügt werden, bis sich der Funktionswert ändert. Die Funktion kann dabei innerhalb gewisser Grenzen frei gewählt werden.

b) Die Einsprungsadresse wird abgesichert durch das Einfügen einer speziellen Bytesequenz, die nicht innerhalb des regulären Codes auftreten kann. Beispiel hierfür ist eine Folge von Op-Codes (SNOP).

Erfindungsgemäß werden also die Einsprungsadressen für Funktionen durch nichtlokale Codekorrelationen, die nur an den Einsprungsadressen auftreten können, abgesichert.

Vorteilhafterweise wird so der Gate Mechanismus, der einen berechneten Jump nach sich zieht und ein Peephole in der Instruktions Pipeline verursacht, vermieden.

Es wird vielmehr direkt an die Einsprungsadresse in die Funktion gesprungen. Die nachfolgenden Instruktionen können in die Pipeline geladen werden, unabhängig davon, ob die Verifikation der Sprungadresse positiv oder negativ ausfällt. Die Effizienz von überwachten Funktionsaufrufen wird damit erhöht.

Patentansprüche

1. Verfahren zur Absicherung von Einsprungsadressen, d a d u r c h g e k e n n z e i c h n e t, daß zulässige Einsprungsadressen direkt angesprungen werden dürfen, und durch eine Korrelation von Daten, die nicht innerhalb der selben, einzelnen Instruktion stehen können, erkennbar sind.

2. Verfahren nach Anspruch 1, d a d u r c h g e k e n n z e i c h n e t, daß der Compiler oder Linker durch Organisation des Programmcodes sicherstellt, daß nur legale Einsprungsadressen diese Korrelation erfüllen.

3. Verfahren nach Anspruch 1 oder 2, d a d u r c h g e k e n n z e i c h n e t, daß die Korrelation dadurch erfolgt, daß die Speicherzelle unmittelbar vor oder nach der Einsprungsadresse die Adresse korrelierter Daten enthält.

4. Verfahren nach Anspruch 3, d a d u r c h g e k e n n z e i c h n e t, daß die Speicherzelle unmittelbar vor oder nach der Einsprungsadresse einen Verweis auf den entsprechenden Eintrag in einer geschützte Liste legaler Einsprungsadressen enthält.

5. Verfahren nach einem der Ansprüche 1 bis 4, d a d u r c h g e k e n n z e i c h n e t, daß beim Ausführen des Funktionsaufrufs automatisch überprüft wird, ob die Korrelation der Daten erfüllt ist.

6. Verfahren nach Anspruch 5, d a d u r c h g e k e n n z e i c h n e t, daß beim Ausführen des Funktionsaufrufs zusätzlich automatisch überprüft wird, ob das korrelierte Datum im vorgesehenen, reservierten Speicherbereich liegt.

7. Verfahren nach Anspruch 1 oder 2, d a d u r c h g e k e n n z e i c h n e t, daß Programminstruktionen eine gewisse maximale Anzahl n von Bytes nicht überschreiten und daß

ein spezieller No-Operation-Code (SNOP) vorgesehen ist, der zur Vermeidung zufälliger Korrelationen dient, und vom Compiler oder Linker nachträglich eingefügt werden kann.

8. Verfahren nach Anspruch 7, d a d u r c h g e k e n n z e i c h n e t, daß die Korrelation zwischen Codedaten erfolgt, die mindestens n Byte voneinander entfernt sind.

9. Verfahren nach Anspruch 1, 2 oder 7, d a d u r c h g e k e n n z e i c h n e t, daß die Einsprungsadresse durch das Einfügen einer speziellen Bytesequenz abgesichert wird, die nicht innerhalb des regulären Codes auftreten kann.

10. Verfahren nach Anspruch 9, d a d u r c h g e k e n n z e i c h n e t, daß als spezielle Bytesequenz ein spezieller No-Operation-Code (SNOP) verwendet wird.

Zusammenfassung

Verfahren zur Absicherung von Einsprungsadressen

Erheblich effizienteres Verfahren zur Absicherung von Einsprungsadressen in Computerprogrammen, wobei zulässige Einsprungsadressen direkt angesprungen werden dürfen, und durch eine Korrelation von Daten, die nicht innerhalb der selben, einzelnen Instruktion stehen können, erkennbar sind. Dabei kann vom Compiler oder Linker durch Organisation des Programmcodes sichergestellt werden, daß nur legale Einsprungsadressen diese Korrelation erfüllen.